AN ELEMENT-BASED SPECTRALLY-OPTIMIZED APPROXIMATE INVERSE PRECONDITIONER FOR THE EULER EQUATIONS

L.E. CARR III , C.F. BORGES , AND F.X. GIRALDO *

Abstract. We introduce a method for constructing an element-by-element sparse approximate inverse (SAI) preconditioner designed to fully exploit the maximum degree of parallelism available in a spectral element modeling environment. This new preconditioning approach is based on a spectral optimization of a low-resolution preconditioned system matrix rather than on a Frobenius norm optimization (FNO) of the full-resolution preconditioned system matrix. We show that the local preconditioning matrices obtained via this element-based, spectrum-optimized (ESBO) approach may be applied to arbitrarily high-resolution versions of the same system matrix without appreciable loss of preconditioner performance. We demonstrate the performance of the EBSO preconditioning approach using 2-D spectral element method (SEM) formulations for a simple linear conservation law and for the fully-compressible 2-D Euler equations with various boundary conditions. For the latter model, the EBSO approach outperforms the FNO approach and, for sufficiently large Courant Number, model wall-clock time is reduced by a factor of 2.

Key words. preconditioning; sparse approximate inverse; element-based; spectral elements; Galerkin methods; Euler Equations

AMS subject classifications. 65M60, 65M70, 35L65, 86A10

1. Introduction.

To provide the motivation behind the preconditioning approach presented herein, we first explain the atmospheric modeling setting within which the preconditioner must perform effectively (Section 1.1). We then provide a formal statement of the preconditioning concept (Section 1.2). In Section 1.3 we summarize most of the commonly used types of preconditioners that are presently available and with an emphasis on their potential strengths/weaknesses relative to our modeling problem, particularly with respect to suitability for use in massively parallel *element-based* computing environments. In Section 1.4 we provide a conceptual overview of the EBSO approach as a prelude to beginning the detailed development in Section 2. In Section 1.5 we describe the organization of the remainder of the paper.

1.1. Modeling Context and Preconditioner Criteria.

A recent paper by Giraldo *et al.* [7] studies a number of semi-implicit (i.e., implicit-explicit, hereafter IMEX) spectral element (SE) formulations of the compressible Navier Stokes equations with a view towards application to nonhydrostatic atmospheric modeling over both regional and global domains [8]. The combination of the high-resolution associated with nonhydrostatic modeling and the large domain size associated with global modeling requires as many as $O(10^7)$ elements and $N_g = O(10^9)$ grid point (nodes). As a result, at each step in the IMEX time integration process there is a need to iteratively solve a very large, but sparse linear system of the form

$$A\boldsymbol{q}^{n+1} = R(\boldsymbol{q}^n) \tag{1.1}$$

where $\boldsymbol{q}$ is a state vector, R is the righthand side operator, and A is a square, invertible, *nonsymmetric* and generally *indefinite*[1] matrix. As discussed in [7, 8], at a minimum the size of A is $N_g \times N_g$ if a *Schur* form is derived for A, and at worst the size of A is $4N_g \times 4N_g$ for 2-D modeling and $5N_g \times 5N_g$ for 3-D modeling if A is left in the non-Schur form.

The motivation for attempting to solve such a challenging problem is that the IMEX time integration approach permits time steps that can be larger than the maximum explicit time step by a factor of 100 or more. As a result, IMEX-based models can actually run faster than models using solely explicit time integrations provided that the number of iterations needed to solve Eq. (1.1) is not too large. As indicated in the conclusion of [7], we are in the process of developing preconditioners for our IMEX models, and our efforts involve both adapting suitable existing methods to our needs, as well as working with new ideas (one of which is reported herein) tailored specifically to our modeling problem.

*Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA 93943, USA

[1]The real spectra shown in Fig. 4.5 of [7] is a special case arising from the combination of using identical square elements and a Schur form for A. The Schur form matrix spectrum becomes complex when other element geometries are employed.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE **An Element-based Spectrally-optimized Approximate Inverse Preconditioner for the Euler Equations**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Applied Mathematics,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
We introduce a method for constructing an element-by-element sparse approximate inverse (SAI) preconditioner designed to fully exploit the maximum degree of parallelism available in a spectral element modeling environment. This new preconditioning approach is based on a spectral optimization of a low-resolution preconditioned system matrix rather than on a Frobenius norm optimization (FNO) of the full-resolution preconditioned system matrix. We show that the local preconditioning matrices obtained via this element-based, spectrum-optimized (ESBO) approach may be applied to arbitrarily high-resolution versions of the same system matrix without appreciable loss of preconditioner performance. We demonstrate the performance of the EBSO preconditioning approach using 2-D spectral element method (SEM) formulations for a simple linear conservation law and for the fully-compressible 2-D Euler equations with various boundary conditions. For the latter model, the EBSO approach outperforms the FNO approach and, for sufficiently large Courant Number, model wall-clock time is reduced by a factor of 2.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**20**	

Given the above modeling context, an effective preconditioned iterative approach to solving Eq. (1.1) must meet the following criteria:

1. Be able to handle a system matrix A that is significantly non-symmetric as well as indefinite.
2. Employ a preconditioner with a processing or application cost that is low in relation to the effectiveness of the preconditioner in reducing the number of iterations so that the wall-clock time for the preconditioned model is significantly smaller than for the unpreconditioned model. Ideally this net speed up should occur when running the model in serial mode (i.e., before parallelization is considered).
3. Employ a preconditioner that admits a very high degree of computing parallelism in general, and that ideally permits using the same SE-based parallelism employed when operating on a vector by the system matrix A.

Taken together, the criteria represent a challenging problem. Criterion 1 rules out the use of the highly efficient conjugate gradient (CG) method, for which convergence is guaranteed only if the matrix is symmetric positive definite (SPD). Of the other interative schemes available to us, we have elected to use the generalized minimum residual (GMRES) method. Moreover, Criterion 1 also essentially rules out whole categories of recent preconditioning work, or at least makes their consideration a lower priority.

Ironically, the need for a low application cost is contrasted by a loose constraint on the initial set-up or construction cost of a preconditioner to be used in our modeling problem. A potentially high construction cost is acceptable because the same preconditioner can be used not only in hundreds of time steps in each model run, but also for potentially *years-worth* of runs in a particular version of an operational weather model (typically run four times a day).

1.2. Preconditioning Problem Statement. We seek to efficiently obtain an approximate solution to the large, sparse linear system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1.2}$$

by iteratively solving either the equivalent left-preconditioned system

$$(KA)\boldsymbol{x} = K\boldsymbol{b} \tag{1.3}$$

or equivalent right-preconditioned system

$$(AK)\boldsymbol{y} = \boldsymbol{b} \qquad K\boldsymbol{y} = \boldsymbol{x}, \tag{1.4}$$

or sometimes a combination of the above two approaches.

In general, the large variety of preconditioning methods in existence can be divided into two classes that have been described as *explicit* and *implicit* [5, p. 970]. If explicit preconditioning is used, then K is a sparse matrix that is designed to approximate A^{-1} in some sense, and thus is often called a sparse approximate inverse (SAI). On the other hand, if implicit preconditioning is being using, then $K = M^{-1}$, where M is a sparse matrix and approximates A. As pointed out in Section 1.3.1, M^{-1} is never actually calculated since it would in general be a full matrix.

If we define the residual vector associated with the i^{th} approximate solution to Eq. (1.2) to be

$$\boldsymbol{r}_i = \boldsymbol{b} - A\boldsymbol{x}_i, \tag{1.5}$$

then an important distinction between Eq. (1.3) and (1.4) is that a left-preconditioned Krylov space algorithm minimizes $K\boldsymbol{r}$, whereas a right-preconditioned algorithm minimizes $\boldsymbol{r}$. Thus, if K is ill-conditioned, then the magnitude of $||\boldsymbol{x} - \boldsymbol{x}_i||$ obtained via a left-preconditioning algorithm may not be accurately reflected by the magnitude of $||K\boldsymbol{r}_i||$. Conversely, if the condition number of K is small, as is the case for problems solved in this paper, then either Eq. (1.3) or Eq. (1.4) can be used, and the choice of which to use simply depends on what works best for a particular problem [12, p. 272]. The results shown in Section 4 are for a *left* EBSO preconditioner since we have found that it provides the best results for the modeling problems addressed herein.

1.3. Overview of Existing Preconditioners. Due to the wide variety and sometimes overlapping nature of preconditioning techniques, overviews of the topic have been organized in various ways by different authors. Here we use the implicit/explicit distinction.

1.3.1. Implicit Preconditioners. For all preconditioners in this class, Eq. (1.4A) becomes

$$A(M^{-1}\boldsymbol{y}) = \boldsymbol{b} \tag{1.6}$$

where to calculate the parenthetical expression during the iterative process we instead solve the sparse system

$$M\widetilde{\boldsymbol{y}} = \boldsymbol{y} \tag{1.7}$$

for the auxiliary vector $\widetilde{\boldsymbol{y}}$.

If the matrix M is a global (i.e., full-grid) incomplete LU factorization (ILU) preconditioner, then Eq. (1.7) becomes

$$LU\widetilde{\boldsymbol{y}} = \boldsymbol{y}, \tag{1.8}$$

where $L + U$ has some specified sparsity pattern, usually based on the sparsity pattern of A [12, p. 287-320]. In a serial computing environment, global ILU is a very powerful preconditioning method owing to the potent combination of the low application cost associated with the forward/backward solves and an often dramatic reduction in the number of GMRES iterations, presumably because the product LU is much less sparse than the sum $L + U$. However, existence of the factorization is not guaranteed even if A is SPD. Moreover for nonsymmetric problems: i) non-existence of the factorization becomes more frequent, ii) factors L and U may be ill-conditioned even if A is well-conditioned, and iii) even if the problems i) and ii) are not encountered the effectiveness of the preconditioner may be much less than that observed if A is SPD [2, p. 436]. In addition, the inherently serial nature of the forward/backward solves offers limited potential for parallelization, and would in any case require a parallelization strategy different from the SE-based approach associated with the system matrix A.

To mitigate the above described issues (particularly the parallelization issue) associated with *global* ILU, various methods that effectively employ multi-level block factorizations (MLBF) have been developed including algebraic multigrid (AMG) and domain decomposition (DD). However, a recent and comprehensive presentation of these state-of-art methods by Vassilevski [14] focuses almost exclusively on SPD systems. In Chapter 8 he does briefly discuss how such methods may be extended to nonsymmetric systems that may be represented as small *perturbations* of symmetric systems. However, for our modeling problem even the banded Schur-form matrices used in [7] have asymmetric components with Frobenius norms of the same order as the symmetric components.

In an effort to achieve optimum parallelization in a SE modeling setting, element-based (i.e., element-by-element or EBE) preconditioners have been developed, usually based on matrix factorization at the element level [12, p. 399]. However, the focus is often on SPD systems [1], possibly to avoid the issue of non-existence of a factored preconditioner for nonsymmetric system matrices. Moreover, an assessment of available EBE preconditioners by van der Vorst in 2002 [15, p.196] indicates that despite their highly parallelizable nature, they have a limited ability to actually reduce model wall-clock time (i.e., application cost basically offsets the potential speedup associated with reduced number of iterations). In a search of the more recent literature we have found nothing to contradict van der Vorst's assessment.

1.3.2. Explicit (SAI) Preconditioners. As mentioned earlier, all SAI preconditioning methods seek to create a sparse matrix K such that $K \approx A^{-1}$. That is, we want K to be "similar" to A^{-1} in some sense. A naive brute-force approach based on *spectral* similarity would be to have a left or right preconditioner K satisfy the respective *global* optimization problems

$$K = \min_{K \in S} \left\| \boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA) \right\|_2 \qquad or \qquad K = \min_{K \in S} \left\| \boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(AK) \right\|_2, \tag{1.9}$$

where S is the set of all matrices satisfying some user-specified sparsity, $\boldsymbol{\sigma}$ is a vector whose components are the eigenvalues of the matrix on which $\boldsymbol{\sigma}$ operates, and each entry of K represents an independent variable to be determined. As it stands, Eq. (1.9) is an intractable problem for very large systems since it necessitates computing the complete spectrum of the preconditioned system matrix (PSM)[2], which is itself an iterative process, at each step of some iterative non-linear least-squares (NLLS) algorithm.

The first tractable alternative to Eq. (1.9) to be developed is the Frobenius norm optimization method (FNO), in which Eq. (1.9) is replaced with the respective *linear* least-squares optimization problems

$$K = \min_{K \in S} \|I - KA\|_F \qquad or \qquad K = \min_{K \in S} \|I - AK\|_F\,. \tag{1.10}$$

In contrast to Eq. (1.9), which attempts to *directly* optimize the spectrum of PSM, the FNO method manipulates the spectrum *indirectly* by making the PSM similar to I in an *entry-by-entry* sense. Despite this indirect approach, the spectrum of an FNO-derived PSM (computable for smaller test systems) typically exhibits a tighter grouping around (1,0) on the complex plane than does the spectrum of A itself and accelerates iterative method convergence.

Although Eq. (1.10) still represents a global optimization problem, it can be recast into a set of N_g independent linear least-squares problems [5, p. 308] that can be solved by direct (*i.e.*, QR) or iterative means [12, p. 323]. Via such methods the entries of K are determined in a column-by-column fashion thus making FNO preconditioner construction tractable for very large systems. Although the construction cost is high compared to an ILU indirect preconditioner, recall that the construction cost is of little concern for a preconditioner to be used in a production-type code. There are triangularly factored forms of SAI preconditioners based on either FNO [9] or incomplete biconjugation[4], and although factorization breakdown can occur, pivoting strategies are available to overcome this problem [3]. The advantage of a factored SAI is that an effectively denser sparsity pattern is achieved just as in the case of implicit ILU preconditioners.

An important advantage of SAI preconditioners is that they are inherently more parallelizable in an SE setting compared to factored implicit preconditioners since their application involves matrix-vector products instead of forward/backward solves. Moreover, in the case of FNO-based preconditioners: i) a solution to Eq. (1.10) always *exists*, and ii) under conditions that are usually met in practice, an FNO-based K can be proved to be non-singular for *any* choice of A that is non-singular [12, p. 325].

Potential issues relative to our modeling problem are: i) the degree of parallelization achievable with SAI is less than optimal from a SE perspective, and ii) thus far the sparsity patterns typically used have not achieved *algorithmic scalability* in that the convergence rate of the preconditioned iterative scheme tends to decrease as the size of the linear system grows [2, p. 424, 464].

A final SAI preconditioner that we will consider is the polynomial preconditioner

$$K = s(A) = \sum_{i=1}^{k} c_i A^i, \tag{1.11}$$

where $s(\ )$ is a low-order Neumann, Chebyshev, or generalized least-squares (GLS) polynomial in A of degree k (usually ≤ 10) [12, pp. 379-388]. Two immediately obvious advantages of polynomial preconditioning are that: i) it is optimally parallelizable from a SE perspective since the parallel computing machinery used to operate by A can also be used to operate by K, and ii) the storage requirements are negligible.

Although a Neumann polynomial preconditioner is not restricted to SPD systems, convergence requires that $A = I - B$ where the spectral radius of B is less than one [15, p. 197], which typically would necessitate the pre-application of another preconditioner to meet the restriction on B. Moreover, any improvement in convergence tends to be largely compensated by increased cost as degree k increases [5, p. 306].

A conceptually desirable aspect of the Chebyshev and GLS polynomial preconditioners is that they strive to make K *spectrally* similar to A^{-1} by replacing Eq. (1.9) with tractable optimization

[2]Hereafter, we will frequently use the acronym PSM to refer to KA and/or AK

problems that manipulate the spectrum of the PSM. A Chebyshev polynomial preconditioner satisfies the L^∞ norm-based minimax problem

$$s(\ ) = \min_{s \in P_k} \left(\max_{\lambda \in E} |1 - \lambda s(\lambda)| \right), \tag{1.12}$$

where P_k is the space of all polynomials of degree $\leq k$ and E is a real interval that encloses $\sigma(A)$. Thus, this method is limited to matrices that have a real spectrum, which does not completely eliminate it from our consideration since, under certain circumstances, the spectrum of the Schur form matrices from our modeling problem are confined to the positive real axis, despite the fact that the matrices are significantly nonsymmetric.

The GLS polynomial satisfies the weighted inner-product norm minimization problem

$$s(\ ) = \min_{s \in P_k} \|1 - \lambda s(\lambda)\|_w \qquad \|p\|_w \equiv \int_E p^2(\lambda) w(\lambda) d\lambda \tag{1.13}$$

where $w(\lambda)$ is a weighting function that is nonnegative over region E on the complex plane, and E encloses the spectrum of A.

In order to employ the Chebyshev and GLS preconditioners an estimate for the spectrum of A must first be obtained. One method for doing this is to embed the polynomial preconditioner within a flexible GMRES scheme and use the spectrum of the upper Hessenberg matrix created during the Arnoldi process as an estimate for the spectrum of A as described in [12, p. 386-9]. Examples of using a GLS polynomial preconditioner to not only reduce the number of GMRES interations, but also achieve wall-clock time reductions for systems of moderate size in a parallel environment (≤ 30 processors) are provided in [10, 11].

1.3.3. Assessment of Existing Preconditioning Methods. Based on the above summary, it appears that the various existing *implicit* preconditioning methods are unsuitable for our purposes since:

- the global ILU-type methods lack sufficient parallelization capability for use with very large systems in a massively parallel computing environment.
- the multi-level (grid) methods (AMG, DD, etc) are presently focussed on SPD systems and the CG iterative method
- available element-based factorizations do not significantly reduce wall-clock time.

However, the existing *explicit* methods based on FNO, biconjugation, and matrix polynomials show significant potential, particularly from the perspective of inherently better parallelizability, and we are currently exploring these options and will report on those results separately. The remainder of this paper provides an initial report on the new explicit EBSO preconditioning method that we are developing.

1.4. Overview of EBSO Preconditioner Concept. In Section 1.3.1 above we notice that the global methods tend to be more effective, but less parallelizable compared to element-based methods, which are optimally parallelizable, but lack effectiveness. A reasonable conjecture is that the global methods are more powerful because in some sense they have a *direct* effect on the spectrum of the global PSM, whereas the element-based methods are directly improving the spectrum of the element matrices, but only *indirectly* (and apparently poorly) improving the spectrum of the global PSM, presumably due to the overlap of the elements matrices as they are effectively assembled in a global matrix. The key idea underlying the EBSO preconditioning approach is to develop an element-based preconditioner (which is optimally parallelizable) via a process that nevertheless directly improves the spectrum of the global PSM.

The EBSO (left) preconditioning approach developed herein begins with revisiting Eq. (1.9) in the following form

$$K = \min_{K \in S} \|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w} \tag{1.14}$$

where the subscripts denote a p-norm that includes a weighting factor w (explained later) that is unrelated to the weighting function in Eq. (1.13). As we have already stated, for the very large A's associated with typical operational models the above problem is intractable as it stands. However, we can transform Eq. (1.14) into a tractable NLLS problem via the following set of observations and assumptions.

Firstly, as in all SE formulations, the *global* system matrix A is effectively assembled from *local* element matrices as indicated by the notation

$$A = \bigwedge_{e=1}^{N_e} A^e \tag{1.15}$$

where N^e is the number of elements and $\bigwedge$ denotes the usual direct stiffness summation (DSS) required by all continuous element-based Galerkin methods. Secondly, based on the observation that in certain types of SE problems many of the A^e matrices have related or even identical entries (e.g., when all elements are the same size and shape), we now assume that a useful global preconditioner K also might be assembled via

$$K = \bigwedge_{e=1}^{N_e} K^e \tag{1.16}$$

where large groups of the elemental matrices K^e's are the *identical.* If fact, we will see that a highly effective global K can be assembled from as few as three unique K^e matrices (for the case of two-dimensions and the same boundary conditions everywhere). By substituting Eqs. (1.15) and (1.16) into Eq. (1.14) we obtain the optimization problem

$$\min_{K \in S} \left\| \boldsymbol{\sigma}(I) - \boldsymbol{\sigma}\left(\left[\bigwedge_{e=1}^{N_e} K^e \right] \left[\bigwedge_{e=1}^{N_e} A^e \right] \right) \right\|_{p,w} \tag{1.17}$$

where the output of the optimization process is a few small matrices that determine K via Eq. (1.16). Computationally speaking, Eq. (1.17) is a significant improvement over Eq. (1.14), since the unknowns we are seeking are the optimal entries for a few small (but potentially full) matrices rather than all the non-zero entries of a global K.

Despite being an improvement over Eq. (1.14), solving Eq. (1.17) is still intractable for large A since it requires us to repeatedly compute the complete spectrum of the global matrix KA inside some NLLS algorithm. The solution to this problem is our observation that:

- if we reduce N^e to a tractable size (e.g., $N^e = 25$) so that a K^e matrix set can be created for a *low-resolution* version of a SE problem
- then we can utilize the same K^e matrix set for a *high-resolution* version of our SE problem (e.g., a 1000-fold increase in N^e)without appreciable loss of preconditioner performance.

1.5. Paper Format. In Section 2 we provide the mathematical basis for the particular global structure of K employed in the EBSO preconditioning approach, and in Section 3 we describe the process by which the local matrices of the EBSO preconditioner are actually computed. In Section 4 we provide and discuss the results of applying the EBSO preconditioning approach. Section 5 concludes the paper with a summary and outline of planned future work in this area.

2. EBSO Preconditioner Structure Formulation.

2.1. 1-D Analysis. The rationale for the chosen structure of the EBSO preconditioner is most clearly demonstrated within the context of using the SE method combined with implicit time differencing to solve a simple 2-D linear conservation law. For reasons that will become clear, we begin with the 1-D linear conservation law

$$\frac{\partial q}{\partial t} + c\frac{\partial q}{\partial x} = 0 \qquad x \in \Omega \tag{2.1}$$

where for our purposes here we can set the basic state advective speed c to unity without any loss of generality. As in all SE formulations, we assume that the domain Ω is composed of elements

$$\Omega = \bigcup_{e=1}^{N_e} \Omega^{(e)} \tag{2.2}$$

and that the physical coordinate (x) and a standard reference coordinate (ξ) are related by the invertible mapping

$$x = f_e(\xi) \qquad \xi \in [-1, +1]. \tag{2.3}$$

If we focus on the e^{th} element and recall that $c = 1$, then Eq. (2.1) becomes

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} = 0 \qquad x \in \Omega^{(e)}. \tag{2.4}$$

Using an N^{th} order Lagrange polynomial basis

$$\{\phi_j(x)\}_{j=1}^{N+1}$$

we can express $q(x, t)$ as

$$q(x,t) = \sum_{j=1}^{N+1} q_j(t)\phi_j(x) + \epsilon_q(x,t) \tag{2.5}$$

where ϵ_q is a residual function arising from approximating $q(x, t)$ using a finite set of basis functions. Substituting Eq. (2.5) into (2.4) results in

$$\sum_{j=1}^{N+1} \frac{dq_j}{dt}\phi_j + \sum_{j=1}^{N+1} q_j \frac{d\phi_j}{dx} = -\frac{\partial \epsilon_q}{\partial t} - \frac{\partial \epsilon_q}{\partial x} \equiv \epsilon(x,t). \tag{2.6}$$

Next, we multiply Eq. (2.6) by any one of the basis functions (*i.e.*, test function) and integrate over the element domain

$$\sum_{j=1}^{N+1} \left(\int_{\Omega^e} \phi_i \phi_j dx \right) \frac{dq_j}{dt} + \sum_{j=1}^{N+1} \left(\int_{\Omega^e} \phi_i \phi_j' dx \right) q_j = \int_{\Omega^e} \phi_i \epsilon dx \qquad i = 1, \ldots, N. \tag{2.7}$$

If we now require the residual function ϵ to be orthogonal to the space spanned by the Lagrange polynomial set, then using matrix notation Eq. (2.7) becomes the normal system

$$M_1^e \frac{d\boldsymbol{q}}{dt} + D_1^e \boldsymbol{q} = \boldsymbol{0} \tag{2.8}$$

where the mass matrix (M_1^e), differentiation matrix (D_1^e), and state vector $\boldsymbol{q}$ are given by

$$M_1^e = \int_{\Omega^e} \phi_i \phi_j dx \qquad D_1^e = \int_{\Omega^e} \phi_i \phi_j' dx \tag{2.9}$$

$$\boldsymbol{q} = [q_1 \ \cdots \ q_{N+1}]^T. \tag{2.10}$$

The subscript denotes that the matrices are associated with a 1-D SE formulation, and are needed to distinguish these matrices from their 2-D counterparts that appear later. Integral formulas equivalent to Eq. (2.9A,B), but expressed with respect to the element reference coordinate, are

$$M_1^e = \int_{-1}^{+1} \phi_i(\xi)\phi_j(\xi) J^e(\xi) d\xi \qquad D_1^e = \int_{-1}^{+1} \phi_i(\xi)\phi_j'(\xi) d\xi \tag{2.11}$$

where $J^e(\xi)$ represents the Jacobian

$$J^e(\xi) = \frac{d}{d\xi} f^e(\xi). \tag{2.12}$$

Generally speaking, the Jacobian is a non-linear function of ξ. However, if there is an affine relationship between the physical and reference coordinates system variables, then the Jacobian is a constant for each element and is simply the ratio of the physical and reference coordinate domains of the element

$$J^e = \frac{\Delta x^e}{2} \qquad \Delta x^e = \texttt{length}\left(\Omega^{(e)}\right). \tag{2.13}$$

In such a case, Eqs. (2.11A,B) can be written

$$M_1^e = J^e \overline{M}_1 \qquad \overline{M}_1 = \int_{-1}^{+1} \phi_i \phi_j d\xi \tag{2.14}$$

$$D_1^e = \overline{D}_1 \qquad \overline{D}_1 = \int_{-1}^{+1} \phi_i \phi_j' d\xi \tag{2.15}$$

where $\overline{M}_1$ and $\overline{D}_1$ are the mass and differentiation matrices for the 1-D reference element, respectively. From Eq. (2.14A) it is clear that for a constant Jacobian the mass matrix of each element is just the reference mass matrix scaled by the Jacobian for that element. From Eq. (2.15A) it is clear that every element differentation matrix is identical to the reference element differentiation matrix since the Jacobian is absent in Eq. (2.11B). These properties will have important implications later on.

Whereas Eq. (2.8) applies to a single element, the analogous form that is defined globally on the domain Ω is

$$M\frac{d\boldsymbol{q}}{dt} + D\boldsymbol{q} = \boldsymbol{0} \tag{2.16}$$

where the global mass and differentiation matrices are obtained via the DSS operations

$$M = \bigwedge_{e=1}^{N_e} M_1^e = \bigwedge_{e=1}^{N_e} J^e \overline{M}_1 \qquad D = \bigwedge_{e=1}^{N_e} D_1^e = \bigwedge_{e=1}^{N_e} \overline{D}_1. \tag{2.17}$$

2.2. 2-D Analysis. With the above 1-D analysis in mind, we now consider the 2-D linearized conservation law analogous to Eq. (2.1), which is

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} = 0 \qquad (x, y) \in \Omega. \tag{2.18}$$

In general, the relationships between the global and elemental domains and the associated physical and reference coordinate systems are given by Eq. (2.2) and

$$x = f_e(\xi, \eta) \qquad y = g_e(\xi, \eta) \qquad \xi, \eta \in [-1, +1] \tag{2.19}$$

$$J_e(\xi, \eta) = \left| \frac{\partial f_e}{\partial \xi}\frac{\partial g_e}{\partial \eta} - \frac{\partial g_e}{\partial \xi}\frac{\partial f_e}{\partial \eta} \right|. \tag{2.20}$$

However, if all elements are rectangular and the physical and reference coordinate systems are again related in an affine manner, then Eqs. (2.19) and (2.20) simplify to

$$x = f_e(\xi) \qquad y = g_e(\eta) \tag{2.21}$$

$$J_e(\xi, \eta) = \left| \frac{\partial f_e}{\partial \xi}\frac{\partial g_e}{\partial \eta} \right| = \frac{\Delta x^e}{2}\frac{\Delta y^e}{2} = J_x^e J_y^e \tag{2.22}$$

where the product of the two Jacobians is simply the ratio of the element area in the physical and reference coordinate systems.

If we again focus on the e^{th} element, then Eq. (2.18) becomes

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} = 0 \quad (x, y) \in \Omega^e. \tag{2.23}$$

We now define a N_2-member multivariate basis

$$\{\psi_k(x, y)\}_{k=1}^{N_2}$$

and expand $q(x, y, t)$ in terms of these 2-D basis functions

$$q(x, y, t) = \sum_{k=1}^{N_2} q_k(t)\psi_k(x, y) + \epsilon_q(x, y, t) \tag{2.24}$$

where again e_q is a residual function. Substituting Eq. (2.24) into (2.23) results in

$$\sum_{k=1}^{N_2} \frac{dq_k}{dt}\psi_k + \sum_{k=1}^{N_2} q_k \frac{\partial \psi_k}{\partial x} + \sum_{k=1}^{N_2} q_k \frac{\partial \psi_k}{\partial y} = -(\epsilon_q)_t - (\epsilon_q)_x - (\epsilon_q)_y \equiv \epsilon(x, y, t). \tag{2.25}$$

As in the 1-D example, we now multiply Eq. (2.25) by any one of the 2-D basis functions and integrate over the element domain, and require the residual to be orthogonal to all the basis functions. The resulting equation set, expressed in matrix notation, is

$$M_2^e \frac{d\boldsymbol{q}}{dt} + D_2^{e,x}\boldsymbol{q} + D_2^{e,y}\boldsymbol{q} = \mathbf{0} \tag{2.26}$$

where the 2-D element mass matrix and differentiation matrices are

$$M_2^e = \int_{\Omega^e} \psi_l \psi_k dxdy \quad D_2^{e,x} = \int_{\Omega^e} \psi_l (\psi_k)_x dxdy \quad D_2^{e,y} = \int_{\Omega^e} \psi_l (\psi_k)_y dxdy \quad k, l = 1, \ldots N_2. \tag{2.27}$$

In order to make use of our earlier 1-D analysis, we rewrite the above matrices for the 2-D conservation law in terms of the matrices for the 1-D conservation law by

- requiring the number of 2-D basis functions N_2 to be $(N+1)^2$,
- writing the 1-D basis functions in terms of the vector $\boldsymbol{\phi}(x) = [\phi_1(x) \ \cdots \ \phi_{N+1}(x)]^T$
- equating each 2-D basis function with the appropriate entry of the outer product of $\boldsymbol{\phi}$ via a column-wise bijective rule for associating the index k with the indices i and j:

$$\begin{bmatrix} \psi_{k=1} & \cdots & \psi_{N(N+1)+1} \\ \downarrow & \cdots & \downarrow \\ \psi_{N+1} & \cdots & \psi_{(N+1)^2} \end{bmatrix} = \begin{bmatrix} \phi_{i=1}(x)\phi_{j=1}(y) & \cdots & \phi_1(x)\phi_{N+1}(y) \\ \vdots & \ddots & \vdots \\ \phi_{N+1}(x)\phi_1(y) & \cdots & \phi_{N+1}(x)\phi_{N+1}(y) \end{bmatrix}. \tag{2.28}$$

It be will convenient to represent the above basis-function-association rule using the concise notations

$$\psi_k = (\phi_i \phi_j)_k = (\phi_i)_k (\phi_j)_k \tag{2.29}$$

where
- the value of the outer index k determines the values of the inner indices i and j
- the index i implies that ϕ depends on x
- the index j implies that ϕ depends on y.

Substituting Eq. (2.29) into (2.27A-C) and exploiting the fact that the double integrals are separable gives:

$$\begin{aligned} M_2^e &= \int_{\Omega^e} (\phi_i)_k (\phi_i)_l dx \int_{\Omega^e} (\phi_j)_k (\phi_j)_l dy \\ D_2^{e,x} &= \int_{\Omega^e} (\phi_i)_k (\phi_i')_l dx \int_{\Omega^e} (\phi_j)_k (\phi_j)_l dy \\ D_2^{e,y} &= \int_{\Omega^e} (\phi_i)_k (\phi_i)_l dx \int_{\Omega^e} (\phi_j)_k (\phi_j')_l dy \end{aligned} \tag{2.30}$$

where $k, l = 1, \dots, (N+1)^2$.

Comparing the right hand sides of Eq. (2.30A-C) with Eq. (2.9A-B) reveals that all entries in the 2-D element mass and differentiation matrices are *Kronecker* products of the 1-D mass and differentiation matrices, which we denote by

$$M_2^e = M_1^e \otimes M_1^e \qquad D_2^{e,x} = D_1^e \otimes M_1^e \qquad D_2^{e,y} = M_1^e \otimes D_1^e. \tag{2.31}$$

By making use of Eq. (2.14A) and (2.15A), we can re-express Eq. (2.31) in terms of the 1-D reference mass and differentiation matrices

$$M_2^e = J_x^e J_y^e \left(\overline{M}_1 \otimes \overline{M}_1\right) \qquad D_2^{e,x} = J_y^e \left(\overline{D}_1 \otimes \overline{M}_1\right) \qquad D_2^{e,y} = J_x^e \left(\overline{M}_1 \otimes \overline{D}_1\right). \tag{2.32}$$

Note that whereas Eq. (2.26) applies to a single element, the analogous form that is defined globally on the domain Ω is

$$M\frac{d\boldsymbol{q}}{dt} + (D_x + D_y)\,\boldsymbol{q} = \boldsymbol{0} \tag{2.33}$$

where the global mass and differentiation matrices above are obtained via the assembly operations

$$M = \bigwedge_{e=1}^{N_e} M_2^e \qquad D^x = \bigwedge_{e=1}^{N_e} D_2^{e,x} \qquad D^y = \bigwedge_{e=1}^{N_e} D_2^{e,y} \tag{2.34}$$

or equivalently, after substituting Eq. (2.32A-C) into Eq. (2.34A-C)

$$M = \bigwedge_{e=1}^{N_e} J_x^e J_y^e \left(\overline{M}_1 \otimes \overline{M}_1\right) \qquad D^x = \bigwedge_{e=1}^{N_e} J_y^e \left(\overline{D}_1 \otimes \overline{M}_1\right) \qquad D^y = \bigwedge_{e=1}^{N_e} J_x^e \left(\overline{M}_1 \otimes \overline{D}_1\right). \tag{2.35}$$

The last step toward identifying the desired structure of the EBSO preconditioner is to discretize Eq. (2.33) in time using a simple two-level implicit time differencing scheme:

$$M\frac{\boldsymbol{q}^{m+1} - \boldsymbol{q}^m}{\Delta t} + (D_x + D_y)\left(\alpha\boldsymbol{q}^m + \beta\boldsymbol{q}^{m+1}\right) = \boldsymbol{0} \qquad \alpha + \beta = 1. \tag{2.36}$$

Solving Eq. (2.36A) for the unknown state vector gives us

$$\left[M + \beta\Delta t\left(D^x + D^y\right)\right]\boldsymbol{q}^{m+1} = \left[M - \alpha\Delta t\left(D^x + D^y\right)\right]\boldsymbol{q}^m \tag{2.37}$$

which has the form of the standard $A\boldsymbol{x} = \boldsymbol{b}$ matrix equation with

$$A = \left[M + \beta\Delta t\left(D^x + D^y\right)\right] \qquad \boldsymbol{b} = \left[M - \alpha\Delta t\left(D^x + D^y\right)\right]\boldsymbol{q}^m. \tag{2.38}$$

Combining Eq. (2.38A) with (2.35), we can express the global system matrix A as

$$A = \bigwedge_{e=1}^{N_e} \left[J_x^e J_y^e \left(\overline{M}_1 \otimes \overline{M}_1\right) + \beta\Delta t J_y^e \left(\overline{D}_1 \otimes \overline{M}_1\right) + \beta\Delta t J_x^e \left(\overline{M}_1 \otimes \overline{D}_1\right)\right] \tag{2.39}$$

where we see that the *global* system matrix A is an assemblage of Kronecker products of *local* 1-D reference mass and differentiation matrices that have been scaled by the Jacobians relating the physical and reference variable coordinate systems. Based on the structure of A that we see in Eq. (2.39), we will now assume that a global preconditioner K constructed according to the rule

$$K = \bigwedge_{e=1}^{N_e} J_x^e J_y^e \left(\overline{K}_1 \otimes \overline{K}_1\right) \tag{2.40}$$

will have the potential to be an effective preconditioner for accelerating the iterative solution to Eq. (2.37). We emphasize here that according to Eq. (2.40), the potentially large global preconditioner is *in principle* based on a single small and local preconditioner matrix, whereas in the *implicit* EBE methods discussed in Section 1.3.1, the local preconditioner matrix is different for every element. However, as discussed in the next subsection, the effect of boundary conditions necessitates that we compute a set of three K_1 matrices, rather than just a single matrix.

In this initial investigation into the EBSO preconditioning method, we will restrict our focus to cases in which all domain elements are of equal size. Under this restriction the Jacobian factors in Eq. (2.40) are the same for all elements, and scale all entries (and thus all eigenvalues) of global preconditioner K to the same degree. As a result, we can incorporate the Jacobian factors into the reference matrices in Eq. (2.40) and assemble a functionally equivalent global preconditioner according to the rule

$$K = \bigwedge_{e=1}^{N_e} (K_1 \otimes K_1)\,. \tag{2.41}$$

We now turn our attention in the next section to the scheme for computing and applying the set of K_1 matrices.

3. EBSO Preconditioner Computation and Application Procedure.

Here we assume that some combination of SE spatial discretization and IMEX time differencing generates the need to iteratively solve the left preconditioned matrix equation

$$KA\boldsymbol{q}^{m+1} = K\boldsymbol{b} \tag{3.1}$$

where the global system matrix A and global preconditioner K are assembled according to Eqs. (2.39) and (2.41), respectively. Based on the structure of K as specified by Eq. (2.41), the spectral optimization problem (1.14) now takes the form:

$$K_1 = \min_{K \in S} \|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w} \tag{3.2}$$

where, were it not for the presence of boundary conditions, a single local K_1 matrix could conceivably be used to construct the global preconditioner K. However, since we must contend with the effect of boundary conditions on the global system matrix A, we must include a way to let the structure of the preconditioner K vary in response to boundary conditions, while at the same time keeping the number of independent variables to be computed to a manageable number. Via some experimentation we have found that a feasible strategy is to limit the number of unique local K matrices to three using the element-based assignment rule

$$K_1 = \begin{cases} K_I & \texttt{if element } e \texttt{ is in the domain } \textit{interior} \\ K_S & \texttt{if element } e \texttt{ includes a domain } \textit{side} \\ K_C & \texttt{if element } e \texttt{ includes a domain } \textit{corner} \end{cases}, \tag{3.3}$$

where we are assuming that the boundary Γ of Ω requires the same set of boundary conditions; if this were not the case then we would have to increase the number of matrices. To simplify the exposition, let us assume that all sides require the same boundary condition.

As a result of Eq. (3.3), Eq. (3.2) is transformed into

$$\{K_I, K_S, K_C\} = \min_{K \in S} \|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w}\,, \tag{3.4}$$

where, if size$(KA) = M$ and λ_k represents an eigenvalue of KA with real and imaginary components represented by $a_k = \mathrm{Re}(\lambda_k)$ and $b_k = \mathrm{Im}(\lambda_k)$, then the weighted p-norm contained in Eq. (3.4) is computed via the formula

$$\|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w} = \left[\sum_{k=1}^{M} \left(|a_k - 1|^p + w\,|b_k|^p \right) \right]^{1/p}. \tag{3.5}$$

The purpose of the factor w is to have the freedom to weight the imaginary components of the spectrum more heavily than the real components. Notice that if we choose $p = 2$ and $w = 1$, then the k^{th} term of the sum in Eq. (3.5) is the square of the distance of the k^{th} eigenvalue from (1,0) on the complex plane.

We use MATLAB's `lsqnonlin` function to iteratively solve Eq. (3.4), after first supplying the algorithm with an initial guess (usually the zero matrix) for matrices K_I, K_S, K_C. As with all NLLS problems, Eq. (3.4) presents challenges with regard to slow convergence rates and non-optimal local extrema. Through experimentation, we have found that letting $p = 4$ has the dual benefit of improving the convergence rate of the NLLS algorithm and improving the convergence rate of GMRES by more tightly clustering the eigenvalues of KA about (1,0) on the complex plane. Again through experimentation, we have found that an adequate NLLS algorithm stopping criterion is a change in relative residual of less than 10^{-3}. To ensure that the NLLS algorithm has found a robust local minimum, we restart the algorithm several times with the values of K_I, K_S, K_C matrices perturbed by 5 percent. The NLLS algorithm running on a desktop PC takes several hours to compute the three EBSO preconditioner matrices.

The power of the EBSO approach is that once the K_I, K_S, K_C matrices have been computed for a low resolution system matrix using the procedure just described, these same matrices are reused without any recomputation for high-resolution systems with many times more elements. That is, regardless of how large N_e becomes, for a rectangular 2-D domains considered in this paper the global preconditioner matrix K is effectively constructed via Eqs. (2.41) and (3.3).

4. Results.

We begin this section by showing how the EBSO preconditioner improves the convergence of GMRES for the simple linear dynamical model on which the structure of the preconditioner was based. We then show and discuss the results of applying the preconditioner to a SE formulation of the non-linear Euler Equations.

4.1. Preconditioning the Linear 2-D Conservation Law Model.

Here we show how the EBSO preconditioner facilitates the solution of Eq. (3.1) where A and $\boldsymbol{b}$ are as given by Eqs. (2.38) and the previous state vector $\boldsymbol{q}_m$ is random vector with normally distributed components supplied by MATLAB's randn( ). We employ trapezoidal implicit time differencing ($\alpha = \beta = 1/2$) with the time-step Δt selected so as to produce a Courant Number of 8, which is large enough to result in an unacceptably slow GMRES convergence rate. We also use 4th-order spatial discretization and doubly periodic boundary conditions on a model domain of $(x, y) \in [0, 10]^2$ meters^2.

To construct a local K_I, K_S, K_C matrix set, we set the number of elements to $N_e = 5 \times 5$ (*i.e.*, in a 5-by-5 grid) and specify that the local preconditioner matrices be assigned to the domain elements according to the pattern shown in Figure 4.1a. We then perform the optimization problem represented by Eq. (3.4) with $p = 4$ and $w = 1$ to obtain local matrices K_I, K_S, K_C. The sparsity pattern for both global matrices A and K appears in Fig. 4.1b. In Fig. 4.1c we provide a comparison of the GMRES convergence rates exhibited for the unpreconditioned (red) and preconditioned (green) problems for a single right-side $\boldsymbol{q}_m$ vector. The significantly more rapid and geometric GMRES convergence provided by the EBSO preconditioner is readily apparent.

In Fig. 4.2 we summarize the results of applying the same EBSO preconditioner used in Fig. 4.1 to a higher resolution version of the conservation law model in which we have increased $N_e = 10 \times 10$, but have reduced Δt sufficiently to maintain the same Courant Number. By comparing Fig. 4.2a with Fig. 4.1a, the reader can see how the same set of local K_I, K_S, K_C matrices are assigned to the larger number of domain elements in the higher resolution model. The representative examples of the GMRES convergence rates (for one of the randomly selected right-side $\boldsymbol{q}_m$ vectors) that we provide in Fig. 4.2c provide a quick visual indication that the EBSO preconditioner created using a lower resolution version of the model still provides a considerable improvement in the GMRES convergence rate for the higher resolution model.

In Table I we provide a statistical summary of how the EBSO preconditioner described above performs in response to changes in Courant No. and number of elements (N_e). By comparing the parenthetical numbers along rows we can see that the reduction in the number of GMRES iterations is relatively insensitive to significant changes in the Courant Number. However, by comparing the parenthetical numbers along columns we can discern a significant and unsatisfactory reduction in the effectiveness of the EBSO preconditioner in response to relatively modest increases in the size of the system matrix.

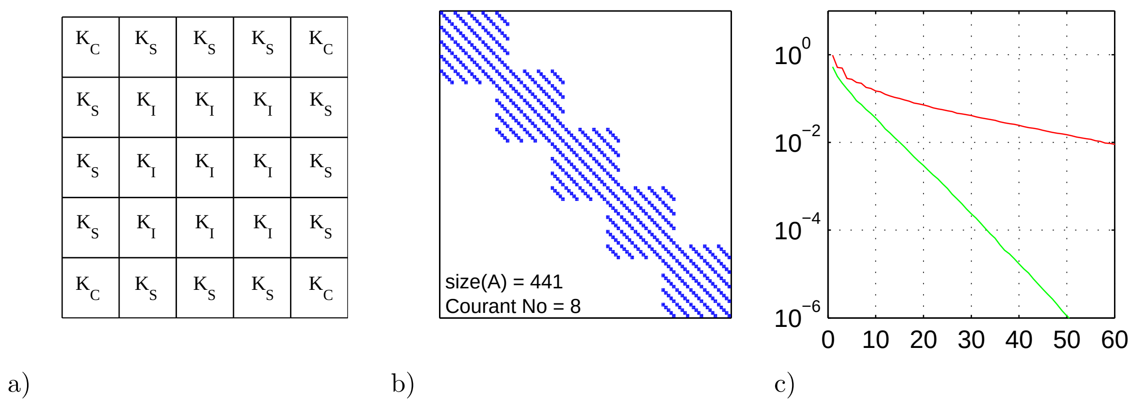

FIGURE 4.1. *Summary of the construction of an EBSO preconditioner for the 2-D linear conservation law using a doubly periodic domain covered by a 5-by-5 grid of elements ($N_e = 5^2$). Figure a)shows the distribution of local preconditioner matrices within the model domain. Figure b) the sparsity pattern of global matrices A and K. Figure c) representative examples of GMRES relative residual versus number of iterations for the unpreconditioned model (red) and the preconditioned model (green).*

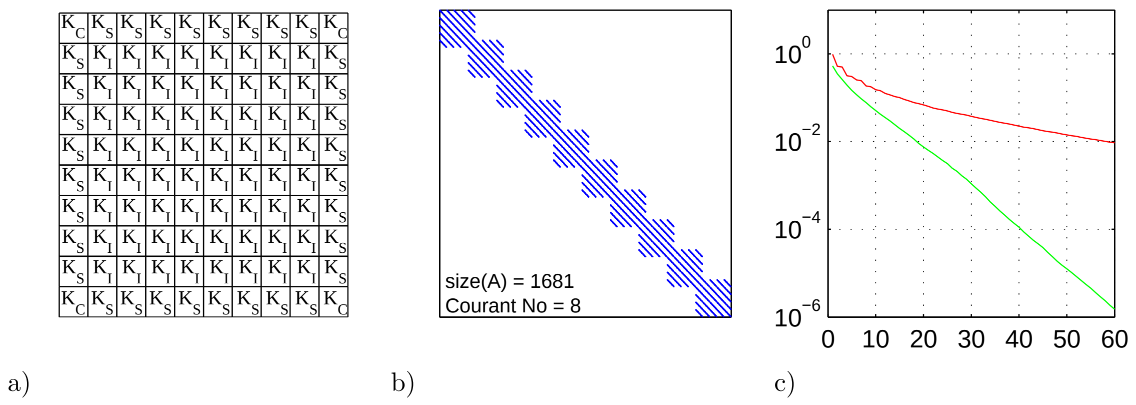

FIGURE 4.2. *As in Fig. 4.1, except that the model domain now consists of a 10-by-10 grid of elements ($N_e = 10^2$).*

To discern the source of the problem just identified, we show in Fig. 4.3 the spectra of the system matrices A and KA corresponding to the Courant No. 8 column of Table I. The linearly-shaped distribution of the spectrum of the matrix A in Fig. 4.3a (red) extends well outside the unit circle and consists of eigenvalues whose arguments range from nearly $-\pi/2$ to nearly $+\pi/2$, and thus covering nearly the entire right half-plane in an *argument* sense. As discussed and demonstrated in [13, p. 271-274; see Fig. 35.4], this eigenvalue property probably explains why the unpreconditioned GMRES convergence rate (Fig. 4.1c; red) is so slow despite the small condition number of A, and thus explains the slow unpreconditioned GMRES convergence rates (red) seen in Figs. 4.1c and 4.2c. By contrast, the disk-shaped spectrum of KA in Fig. 4.3a (green) is confined well within the unit circle and exhibits a much more restricted range of arguments for the majority of eigenvalues. As discussed and demonstrated in [13, Fig. 35.2], this argument property is consistent with the faster preconditioned GMRES convergence rate seen in Fig. 4.1c (green).

However, notice in Figs. 4.3b and c that as model resolution is increased the spectra of KA develop an increasingly dense, *vertically-oriented* line of eigenvalues that extends outside the unit circle. Thus, it appears that a growing number of eigenvalues acquire increasingly large imaginary components as the size of the matrix KA increases, which we conjecture is the source of the degradation in the preconditioned GMRES convergence rate with increasing model resolution.

Based on the above conjecture, and to overcome the presumably deleterious vertical spreading of the spectrum of KA as the matrix size is increased, we increased the imaginary component weighting

	Courant Number		
N_e and Size(A)	8	16	24
5^2 441	57.4 16.3 (3.52)	80.0 22.7 (3.53)	90.3 27.0 (3.35)
10^2 1681	57.8 19.6 (2.95)	82.2 22.3 (3.01)	93.8 32.6 (2.88)
15^2 3721	57.8 20.9 (2.76)	81.7 30.5 (2.68)	93.4 36.6 (2.55)

TABLE I

Average number of unpreconditioned (upper), preconditioned (lower) GMRES iterations and their quotient (in parentheses) for the linear conservation law as a function of number of elements (N_e) and Courant Number. Results are based on a sample size of 20, a stopping relative residual of 10^{-2}, and the EBSO preconditioner shown in Fig. 4.1.

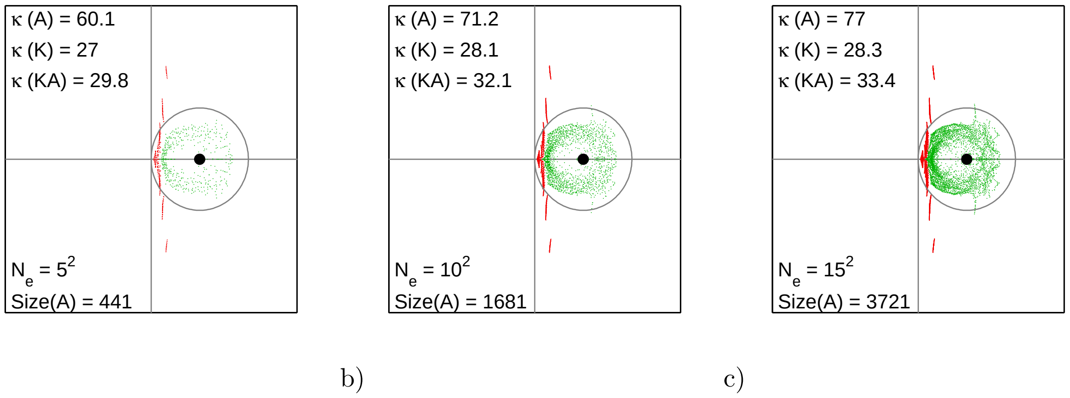

FIGURE 4.3. *Spectra of the global system matrices A (red) and KA (green) resulting from Courant No. of 8 and number of elements shown in the lower left of each panel. The condition numbers of matrices A, K, and KA appear in the upper left of each panel.*

factor to $w = 10^3$ in the optimization problem (3.4), and then created a new set of local K_I, K_S, K_C matrices, again using a low model resolution of $N_e = 5^2$. This change results in the quasi-elliptical spectrum distribution for the matrix KA shown in Fig. 4.4a, in which the majority of eigenvalues (particularly the larger ones) have imaginary components that are small compared to their real components.

Now notice that as we increase the number of elements to $N_e = 10^2$ (Fig. 4.4b) and $N_e = 15^2$ (Fig. 4.4c), while again using the same set of local preconditioner matrices K_I, K_S, K_C, the spectrum expands only slightly in the imaginary direction. In Figs. 4.4d-f we show that a similar behavior occurs when the same preconditioner is applied to various model resolutions with the time step increased to give a Courant No. of 16. Notice that although the spectra extend outside the unit circle in the direction of the positive real axis in Fig. 4.4d-f, nevetheless the shape of the spectra retains the same sort of elliptical character as seen in Fig. 4.4a-c, and thus maintains a consistently tight restriction on the magnitude of the imaginary components relative to the real components for the larger eigenvalues. We note in passing that in Fig. 4.4a-c, for the which the Courent No. is fixed at 8, the distribution of the spectrum of A is visually virtually unchanged in Fig. 4.4a compared to Fig. 4.4c. The insensitivity of the spectral distribution of the Schur-form matrix to increases in system size (model resolution) probably plays a key factor in the insensitivity of the EBSO preconditioner to changes in system size.

In Table II we summarize the performance of the above EBSO preconditioner created via a weighted NLLS optimization process. By comparing the parenthetical numbers along columns we can see that there is now no significant decrease in preconditioner performance as the number of elements is increased. By comparing the parenthetical numbers along rows we can see that there is

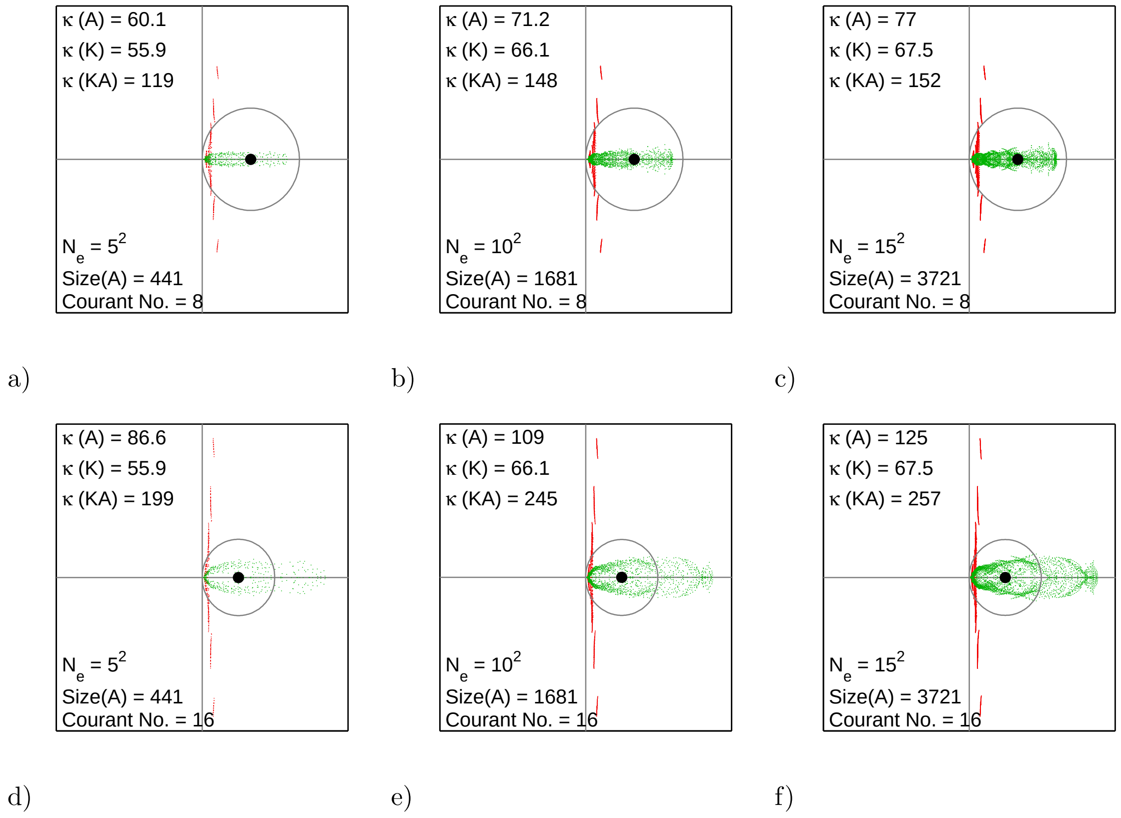

Figure 4.4. *Comparison of the spectra of the preconditioned (green) system matrices as the number of elements is increased for a Courant No. of 8 (Panels a-c) and Courant No. of 16 (Panels d-f).*

		Courant Number		
N_e and Size(A)		8	16	24
5^2	441	57.5 16.5 (3.48)	79.3 21.4 (3.71)	89.8 24.0 (3.73)
10^2	1681	58.2 17.0 (3.42)	83.0 22.1 (3.75)	94.0 25.1 (3.75)
15^2	3721	58.2 17.3 (3.37)	81.8 22.1 (3.70)	93.1 24.9 (3.74)

Table II

As in Table I, except for the EBSO preconditioner described by the text associated with Fig. 4.4.

now actually a tendency for preconditioner performance to improve as the Courant No. is increased.

4.2. Preconditioning the Non-Linear 2-D Euler Equations. Here we show how the EBSO preconditioner facilitates the solution of a particular formulation of the 2-D Euler equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \qquad \frac{\partial \theta}{\partial t} + \boldsymbol{u} \cdot \nabla \theta = 0$$
$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \frac{1}{\rho} \nabla P + g\boldsymbol{k} = 0 \qquad P = P_A \left(\frac{\rho R \theta}{P_A} \right)^{\gamma} \tag{4.1}$$

where ρ is the density, θ is the potential temperature, $\boldsymbol{u} = (u, w)^T$ is the velocity field, P is the pressure, P_A is the surface pressure, R is the ideal gas constant, and $\gamma = 1.4$. After partitioning the state variables into a hydrostatic basic state and a non-hydrostatic perturbation, Giraldo *et al.* [7] present a detailed methodology for development of a SE model employing semi-implicit time integration and utilizing a Schur complement form of Eq. (4.1) so that the perturbation state variables may be obtained in a sequential fashion beginning with the pressure.[3]

The three dynamical test cases (i.e., inertia gravity wave, density current, and mountain wave) employed in [7] to evaluate the performance of the model did not result in a large number of GMRES iterations, and thus did not necessitate preconditioning. Therefore, we will employ a fourth standard test case here that does result in an excessive number of unpreconditioned GMRES iterations; namely, a rising thermal bubble [6]. The problem domain is $(x,z) \in [0,1000]^2$ meters^2 with reflecting boundary conditions on all four sides. The basic state is a motionless air mass that has a potential temperature of 300^{o}K and that is in hydrostatic balance. The initially motionless bubble has a perturbation potential temperature θ' given by:

$$\theta' = \begin{cases} 0 & \texttt{if } r > r_c \\ \frac{\theta_c}{2}\left[1+\cos\left(\frac{\pi r}{r_c}\right)\right] & \texttt{if } r \le r_c \end{cases} \qquad r = \sqrt{(x-x_c)^2+(z-z_c)^2} \tag{4.2}$$

where $\theta_c = 0.5^o$C, $r_c = 250$m, $(x_c, z_c) = (500{,}350)$. For all the results that follow we employ 5th-order spatial interpolation. We also use the same 10^{-2} relative residual as the stopping criterion for GMRES, and the same 2nd-order backward time differencing scheme (BDF2) and serial computing environment as used in [7].

Figure 4.5 provides a visual comparison of the evolution of the bubble out to 650 seconds for a short time step reference run (Fig. 4.5a), a large time step unpreconditioned run (Fig. 4.5b) and large time step EBSO preconditioned runs (Fig. 4.5c). Notice that the warmest potential temperature pertubation found anywhere (T_{max}) at $t = 650s$ in the two large time step runs is the same to 6 decimal places, which indicates that the EBSO preconditioner introduces no significant error to the GMRES computations. That T_{max} in the two large time step runs varies from the short time step reference run by 1 percent is consistent with the 10^{-2} relative residual stopping criterion for GMRES, and confirms that the BDF2 time-differencing scheme can handle the very long time step (≈ 100 times the maximum explicit time step) associated with a model run at Courant No. = 28.

To construct the preconditioner employed in the rising bubble problem we again used $N_e = 5^2$, $p = 4$, and $w = 10^3$ based on the results in Section 4.1. However, this time we set the Courant Number at 18 to obtain a similar unpreconditioned GMRES convergence rate as in Section 4.1. Figure 4.6a-d shows the results when the K_I, K_S, K_C matrices are applied to a larger $N_e = 10^2$ problem. In contrast to the linear conservation law problem (in which the sparsity patterns of K and A are the same), here the two sparsity patterns are different. Notice that the bandwidth of the Schur form system matrix (Fig. 4.6b) is twice that of the global preconditioner (Fig. 4.6a) because the

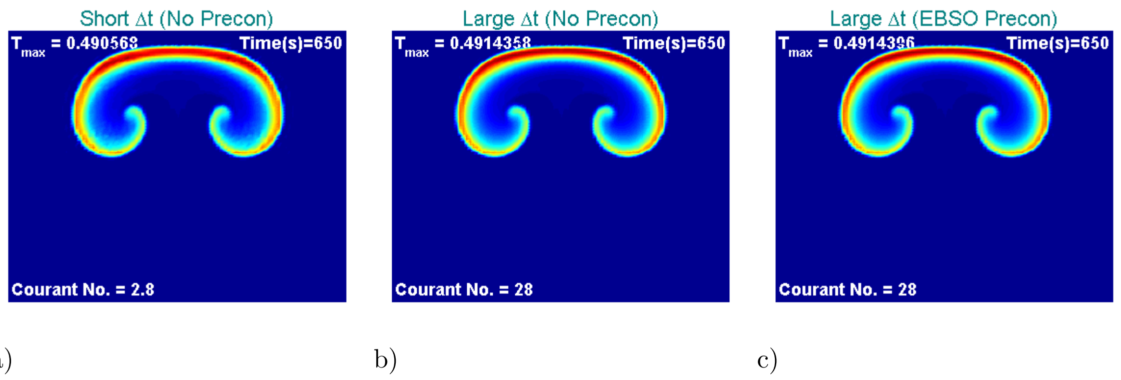

FIGURE 4.5. *Perturbation potential temperature fields at 650s from rising thermal bubble problem using a small Δt (panel a), a large Δt and no preconditioner (panel b), and the same large Δt and the EBSO preconditioner. For all runs $N_e = 40^2$, and the T_{max} number in the upper left of each figure gives the warmest temperature at any node in the domain.*

[3]For those readers interested in more details, see the portion of [7] dealing with Eq. set (4.1) to which they assign the label SE2NC.

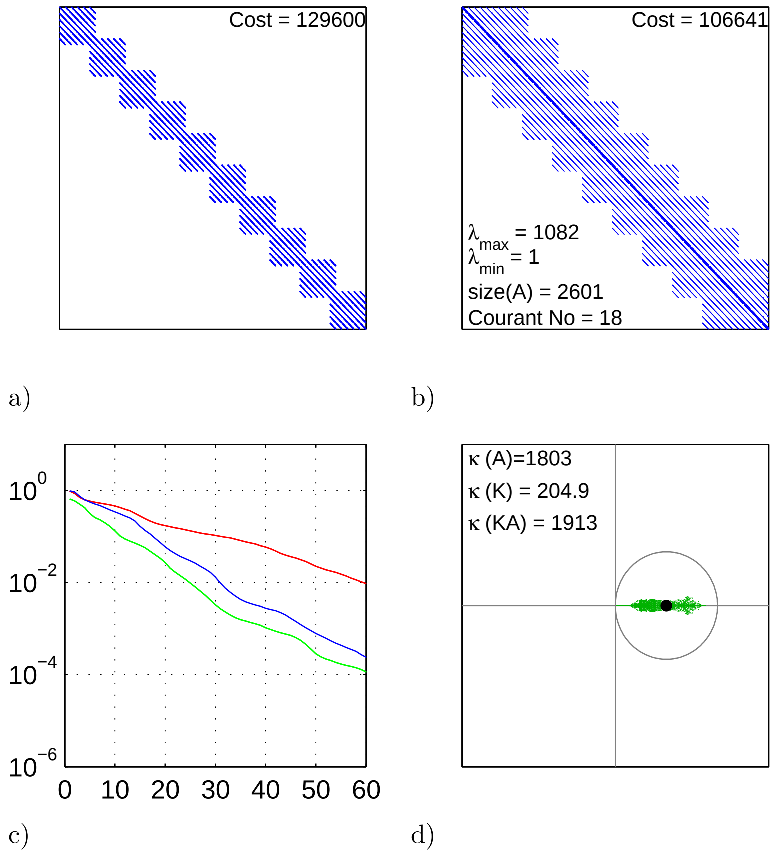

FIGURE 4.6. *Results of applying an EBSO preconditioner (created using Ne = 5x5) to a non-hydrostatic rising thermal bubble case for a higher version of the model using Ne = 10x10. The panels show: a) the sparsity pattern and approximate AC of the global matrix K; b) the sparsity pattern, approximate AC, and specifications for the global matrix A; c) Representative examples of GMRES convergence rates using no preconditioner (red), the EBSO preconditioner (green), and a FNO preconditioner (blue) based on the same sparsity pattern seen in panel a); and d) the spectrum of the global matrix KA and the boundary of the unit disk centered at unity.*

Schur form involves a product of discretized operators (i.e., divergence × gradient). Nevertheless, the approximate application cost to multiply by K is 22 percent more than for the matrix A. These differences are explained by the fact that the Schur form sparsity pattern is relatively less dense due to the use of collocated (i.e., inexact) quadrature, whereas the EBSO preconditioner employs a relatively dense sparsity pattern equivalent to that which would arise from computing a mass matrix using exact quadrature. The justification for the different sparsity pattern is simply pragmatic: the narrower, but denser sparsity pattern for K is much easier to construct and performs significantly better than an EBSO preconditioner employing the Schur form sparsity pattern. Thus, we have here another example of the familiar fact that the sparsity of A may not provide the most powerful K.

Whereas the spectrum of the system matrix A is real with the maximum and minimum values listed in Fig. 4.6a, the spectrum of global KA (Fig. 4.6d) exhibits an elliptical shape roughly centered on unity similar to that seen in Fig. 4.4a. The EBSO preconditioned GMRES convergence rate (Fig. 4.6c; green) is a marked improvement over the unpreconditioned convergence rate (Fig. 4.6c; red), and is a moderate improvement over the convergence rate of a standard FNO preconditioner (Fig. 4.6c; blue), which we constructed using the sparsity pattern of the Schur form system matrix (Fig. 4.6b). We emphasize here that all computation in this initial analysis was done in a serial mode. Since the EBSO is a scalable (see Tables III and IV) and optimally parallelizable (in an SE setting) preconditioner and the FNO preconditioner has less parallelization potential, we expect that the EBSO preconditioner will significantly outperform the FNO preconditioner in a highly parallel

	Courant Number			
N_e and Size(A)	8	16	24	32
25^2 $15,876$	18.69 11.34 (1.65)	36.75 18.92 (1.94)	52.74 25.51 (2.06)	68.09 31.62 (2.15)
50^2 $63,001$	16.85 10.14 (1.66)	36.89 18.24 (2.02)	56.83 26.49 (2.15)	73.05 33.68 (2.17)
100^2 $251,001$	15.62 9.588 (1.63)	33.72 17.13 (1.96)	52.32 25.37 (2.06)	73.40 32.97 (2.22)
200^2 $1,002,001$	15.66 9.382 (1.68)	31.48 16.61 (1.89)	48.09 24.51 (1.96)	67.64 32.02 (2.11)

Table III

Average number of unpreconditioned (upper) and preconditioned (lower) GMRES iterations and their quotient (parentheses) as a function of number of elements and Courant Number for the rising bubble problem integrated for 100 s. The sample sizes on which the averages are based vary as a function of resolution and Courant No. ranging from a minimum of 163 for $N_e = 25^2$ and Courant No. = 32 to a maximum of 5226 for $N_e = 200^2$ and Courant No. = 8.

	Courant Number			
N_e and Size(A)	8	16	24	32
25^2 $15,876$	42.33 37.83 (1.12)	44.16 28.20 (1.57)	47.66 25.43 (1.87)	50.72 23.88 (2.12)
50^2 $63,001$	483.0 386.8 (1.24)	515.0 312.8 (1.65)	565.8 296.9 (1.90)	581.5 284.5 (2.04)
100^2 $251,001$	3971 3208 (1.24)	4114 2566 (1.60)	4464 2463 (1.81)	5036 2393 (2.10)
200^2 $1,002,001$	35287 26338 (1.34)	37731 21899 (1.72)	43360 21822 (1.98)	53132 22244 (2.39)

Table IV

As in Table III, except that the table entries are wall clock time in seconds.

computing environment.

In Tables III and IV we provide a statistical summary of the performance of the EBSO preconditioner for the rising bubble problem with regard to the number of GMRES iterations and model wall-clock time, respectively. All the model runs used to construct the tables were limited to 100 seconds of integration time, since it is in the earlier phase of the rising bubble problem for which the number of GMRES iterations are the largest, and thus for which preconditioning is most needed.

By comparing the parenthetical numbers in Table III along rows we can see that as the Courant No. is increased there is a modest tendency for improvement of preconditioner performance in terms of reducing the relative number of GMRES iterations. By comparing the parenthetical numbers along columns we can see that there is no significant change in preconditioner performance as the number of elements is increased to as much as 1600 times the number of elements used to construct the local preconditioner matrices K_I, K_S, K_C. This is a pivotal result and suggests the effectiveness of the EBSO preconditioner in reducing the number of GMRES iterations is, for all practical purposes, independent of model resolution for the rising bubble problem.

By comparing the parenthetical numbers in Table IV along columns we can see that the factor by which model wall-clock time is reduced is approximately constant, and if anything, shows a slight tendency to increase as the size of the system matrix increases. By comparing the parenthetical number in Table IV along rows we can see that the wall-clock time reduction factor improves significantly as Courant No. is increased.

A very important final point to emphasize concerning Table IV has to do with *actual* wall-clock time. Notice that the wall-clock time for the unpreconditioned model runs (upper entry in each cell) increases significantly as the Courant No. (and thus time step) is increased, which tends to offset the advantage of long time-step available to the implicit integrator. By contrast the wall-clock time for the preconditioned model runs (lower entry in each cell) decreases significantly as the Courant No. (and thus time step) is increased (except for the $N_e = 200^2$, Courant No. = 32 combination), which tends to enhance the advantage of long time-step available to the implicit integrator. For example, for $N_e = 100^2$ the wall-clock time for the unpreconditioned model increased from 3971 to 5036 seconds as Courant No. increased, whereas for the preconditioned model wall-clock times decreased from 3208 to 2393 seconds.

5. Summary and Concluding Remarks. We have provided an initial development and demonstration of a preconditioning method for accelerating the iterative solution of the system $A\boldsymbol{x} = \boldsymbol{b}_i$ that arises at the i^{th} time step in any spectral element method (SE) fluid dynamics model that employs semi-implicit time integration. By means of this preconditioning method the large and sparse global approximate inverse preconditioner K in the equivalent left-preconditioned system

$$(KA)\boldsymbol{x} = K\boldsymbol{b}_i \tag{5.1}$$

is effectively assembled from a set of a few small and full local matrices whose entries result from an optimization problem that seeks to make the matrix KA the best approximation of I in an eigenvalue-by-eigenvalue sense (i.e., spectrally) rather than in an entry-by-entry sense as in the Frobenius norm method. Thus, we have assigned the preconditioning method the descriptor "element-based, spectrum-optimized" (EBSO).

Using both a 2-D linear conservation law and the 2-D fully compressible, non-linear Euler equations, we have demonstrated that after the local matrices of the EBSO preconditioner are created using a low resolution version of the applicable SE model, the preconditioner then may be applied to arbitrarily high-resolution versions of the same model without appreciable loss of preconditioner performance. In a test case in which the Euler equations are used to model a rising thermal bubble, the number of GMRES iterations is cut in half for Courant numbers of 16 and greater, and the model wall-clock time cut approximately in half for Courant numbers of 24 and greater.

The EBSO preconditioner we have introduced has essentially the same parallelization potential in an SE computing enviroment as the unpreconditioned system matrix since no computing machinery other than that needed to operate by A at each time step is needed to operate by K at each time step. Moreover, the EBSO preconditioner requires significantly less storage than exisiting EBE preconditioners, which require the storage of one or more unique local matrices for each domain element. By contrast, only a few local (e.g., $N+1$ by $N+1$ when using N^{th}-order spatial discretization) preconditioner matrices need be stored on each *processor*, and only one local matrix *i.e.*, K_I needs to be stored on the many processors that are assigned to only interior elements. The preconditioner does have a relatively high construction cost, but since the eventual target application is operational weather prediction models that run for years without modification, and involve hundreds of time-steps during each run of the model.

In terms of application, our future work will include extending the EBSO methodology to include variable domain elements sizes and geometries, 3-D modeling using parallel computing, and developing preconditioners for other boundary conditions and other implicit time-differencing schemes such as semi-implicit Runge-Kutta methods.

Acknowledgements. The authors gratefully acknowledge the support of the Computational Mathematics program of the Air Force Office of Scientific Research. We also thank Jim Kelly for providing thorough comments to the manuscript and providing many helpful suggestions.

REFERENCES

[1] C.E. Augarde, A. Ramage, and J. Stauchacher, *An element-based displacement preconditioner for linear elasticity problems*, Computers and Structures, 84 (2006) pp. 2306-2315.

[2] M. Benzi, *Preconditioning Techniques for Large Linear Systems: A Survey*, J. Comput. Phys., 182 (2002), 418-477.

[3] M. Benzi, J.C. Haws, and M. Tuma, *Preconditioning Highly Indefinite and Nonsymmetric Matrices*, SIAM J. Sci. Comput., 22 (2000) pp. 1333-1353.

[4] M. Benzi and M. Tuma, *A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., 19 (1998) pp. 968-994.

[5] M. Benzi and M. Tuma, *A Comparative Study of Sparse Approximate Inverse Preconditioners*, Appl. Numer. Math., 30 (1999) pp. 305-340.

[6] F.X. Giraldo and M. Restelli, *A Study of Spectral Element and Discontinuous Galerkin Methods for the Navier-Stokes Equations in Nonhydrostatic Mesoscale Atmospheric Modeling: Equation Sets and Test Cases*, J. Comp. Phys., 227 (2008), pp. 3849-3877.

[7] F.X. Giraldo, M. Restelli, and M. Lauter, *Semi-Implicit Formulations of the Navier-Stokes Equations: Applications to Nonhydrostatic Atmospheric Modeling*, SIAM J. Sci. Comput., 32 (2010), pp. 3394-3425.

[8] J. F. Kelly and F.X. Giraldo, *Development of the Nonhydrostatic Unified Model of the Atmospheric (NUMA): Limited Area Mode*, J. Comp. Phys., submitted.

[9] L.Y. Kolotilina and A. Y. Yeremin, *Factorized Sparse Approximate Inverse Preconditioning. II: Solution of 3D FE Systems on Massively Parallel Computers*, Int. J. High Speed Comput., 45 (1993).

[10] Y. Liang, *Generalized Least-Squares Polynomial Preconditioners for Symmetric Indefinite Linear Equations*, Parallel Comput., 28 (2002), 323-341.

[11] Y. Liang, *Polynomial Preconditioner for the Solution of Linear Equations*, Ph.D. dissertation, University of Ulster, 2005.

[12] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia PA, 2003.

[13] L.N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia PA, 1997.

[14] P. S. Vassilevski, *Multilevel Block Factorization Preconditioners*, Springer, New York NY, 2008.

[15] H.A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, New York NY, 2003.